[illegible]

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/760,031	01/12/2001	Robert H. Halstead JR.	09612.1014-02000	1846

22852 7590 06/15/2005

FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER LLP
901 NEW YORK AVENUE, NW
WASHINGTON, DC 20001-4413

EXAMINER	
KANG, INSUN	
ART UNIT	PAPER NUMBER
2193	

DATE MAILED: 06/15/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 09/760,031	Applicant(s) HALSTEAD ET AL.
	Examiner Insun Kang	Art Unit 2193

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 4/13/05,12/6/04,6/24/04, and 2/2/04.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-25 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-25 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on 06 December 2004 is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date 4/13/05,12/6/04,2/2/04

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the amendment filed 4/13/2005, 12/6/2004, 6/24/2004, and 2/2/2004.

2. As per applicant's request, claims 1, 11, 21, 22, and 25 have been amended. Claims 1-25 are pending in the application.

Double Patenting

3. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970);and, *In re Thorington,* 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent is shown to be commonly owned with this application. See 37 CFR 1.130(b).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

4. Claims 1-25 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting over claims 1-10, 13-22 and 25-28 copending Application No.09/759697.

Although the conflicting claims are not identical, they are not patentably distinct from each other because they are directed to substantially the same invention and recites only obvious differences which would have been obvious to one of ordinary skill in the art of program development at the time of invention such as simply (i) omitting/adding steps or elements along with their functions, and/or (ii) implementing the method steps with means for performing the steps, and/or (iii) computer program implementation of the method, and/or (iv) implementing a system, product and data signal having computer program for performing the method steps, as explained below.

The corresponding claims are as follows:

Instant claim:	copending claims:
1	1, 10, 13,22, 25,26,28
2	2, 14
3	3, 15
4	1, 13
5	4, 16
6	5, 17
7	6, 18
8	7, 19, 27
9	8, 20
10	9, 21
11	1;10, 13,22, 25,26,28
12	14, 2
13	15; 3
14	13, 1
15	16, 4
16	17, 5
17	18, 6
18	19, 7
19	20, 8
20	21, 9
21	1,10, 13,22, 25,26, 28
22	1,10, 13,22, 25,26, 28
23	6, 18
24	27, 19, 7
25	1,10, 13,22, 25,26, 28

Per claim 1:

Copending claim 1 recites **a method of processing data comprising: defining an object with defined with an option data structure which supports references to option values without preallocation of memory space for the full option values** ("A method of processing data comprising: defining a class which supports an option data structure having, in instances of the class, references to option values without preallocation of memory space for the full option values"), as recited in instant claim 1. The copending claim 1 does not explicitly recite the additional limitations, as further recited in instant claim 1:

i) **defining an object with defined fields to support values in preallocated memory space** as recited in instant claim 1, but Official Notice is taken that this step was well known in the art of object oriented program development at the time of invention was made: A class definition typically includes one or more fields that declare and store data of various types. The compiler assigns a memory location (preallocate) to each field in the program based on its type accordingly and then the attribute value of the field is stored in the memory space preallocated to that field -- each instance of a class (object) has its own copy of the fields defined in the class. Therefore, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the step recited in the co-pending claim by adding the step of defining object with defined fields for the purpose of defining and storing data in the memory space preallocated by the compiler. Alternatively, the omission of defined fields from the instant method would be obvious because option values for the purpose of expediting the method.

Ii) **accessing a field value and accessing an option value in the object using expressions of the same syntactic form**, as further recited in instant claim 1. The co-pending claim 1 does not explicitly recite this step. However, both field values and option values are property values of an instance object or a class. Official Notice is taken that this definition was well known in the art of object oriented program development at the time of invention was made: in object oriented programming, properties are usually made as private so that only member methods have access to the properties for purpose of encapsulation. Accessors (getters/setters) are typical of a well-designed object used for accessing a property of a class/object. Regardless of the different memory allocation methods of option values and field values, the use of the same syntax would be obvious because only member methods can access both option and field values. For example, the member method call syntax is objectName.methodName() in Java or C++. The dot operator is used between class/object name and method name: objectName.getValue(); objectName.setValue(field, option); objectName.fieldValue; objectName.optionValue. It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the step recited in the co-pending claim 1 by adding the step of **accessing a field value and accessing an option value in the object using expressions of the same syntactic form,** as recited in instant claim 1 because both option and field values are properties of the object.

The instant claim does not explicitly recite the step of **during compilation, using the type description in the option data structure to process an operation**

on the option value, as recited in co-pending claim 1. It would have been obvious for one of ordinary skill in the art of program development at the time the instant invention was made to modify the co-pending method by omitting the step of **during compilation, using the type description in the option data structure to process an operation on the option value** recited in co-pending claim 1 for the purpose of expediting the method.

Per claim 2:

The rejection of claim 1 is incorporated, and further, the instant claim recites the additional limitation regarding change handler code which corresponds to co-pending claim 2.

Per claim 3:

The rejection of claim 2 is incorporated, and further, the instant claim recites the additional limitation regarding change handler code for one option defined in different classes within a class inheritance hierarchy and from each class executed when the option value changes which corresponds to copending claim 3.

Per claim 4:

The rejection of claim 1 is incorporated, and further, the instant claim recites the additional limitation regarding the type description in the option data structure to process an operation on the option value which corresponds to copending claim 1.

Per claim 5:

The rejection of claim 1 is incorporated, and further, the instant claim recites the additional limitation regarding getting the set option value and getting the default value for the class which corresponds to copending claim 4.

Per claim 6:

The rejection of claim 1 is incorporated, and further, the instant claim recites the additional limitation regarding defining a first class and second class and encoding an option operation which corresponds to copending claim 5.

Per claim 7:

The rejection of claim 1 is incorporated, and further, the instant claim recites the additional limitation regarding notifying objects of a change in an option value and the option binding which corresponds to copending claim 6.

Per claim 8:

The rejection of claim 1 is incorporated, and further, the instant claim recites the additional limitation regarding a linked list of option items which corresponds to copending claim 7.

Per claim 9:

The rejection of claim 1 is incorporated, and further, the instant claim recites the additional limitation regarding nonlocal option hierarchy which corresponds to copending claim 8.

Per claim 10:

The rejection of claim 9 is incorporated, and further, the instant claim recites the additional limitation regarding graphical hierarchy which corresponds to copending claim 9.

Per claims 11-21:

The rejection of claims 1-10 is respectively incorporated, and further, the instant claims recite a system corresponding to copending claims 13-18, 20-22 and 25 respectively, modified in the manner set forth above in connection with claims 1-10 respectively. It would have been obvious for one of ordinary skill in the art of program development to implement the copending method modified in the manner set forth above with a system including means for performing the steps of the copending method.

Per claim 22:

The rejection of claims 1-21 is respectively incorporated, and further, the instant claim recites a program product corresponding to copending claim 26, respectively, modified in the manner set forth above in connection with claims 1-21 respectively. It would have been obvious for one of ordinary skill in the art of program development to implement the copending method modified in the manner set forth above with a program product including means for performing the steps of the copending method.

Per claim 23:

The rejection of claims 1, 7, 11, 17 and 22 is respectively incorporated, and further, the instant claim recites the additional limitation regarding instructions to notify objects of a change in an option value which corresponds to the step of copending claims 6 and 18, except that the instant claim does not explicitly recite how to notify objects. It would have been obvious for one of ordinary skill in the art of program development at the time the instant invention was made to modify the copending method by omitting the method of notifying objects in copending claims 6 and 18 for the purpose of expediting the method.

Per claim 24:

The rejection of claims 1, 8 and 18 is respectively incorporated, and further, the instant claims recite a program product corresponding to copending claim 27, respectively, modified in the manner set forth above in connection with claims 1, 8 and 18 respectively. It would have been obvious for one of ordinary skill in the art of program development to implement the copending method modified in the manner set forth above with a program product including means for performing the steps of the copending method.

Per claim 25:

The rejection of claims 1, 11, 21 and 22 is respectively incorporated, and further, the instant claims recite a data signal corresponding to copending claims 28 respectively, modified in the manner set forth above in connection with claims 1, 11, 21 and 22 respectively. It would have been obvious for one of ordinary skill in the art of program development to implement the copending method modified in the manner set forth above with a data signal including means for performing the steps of the copending method.

This is a provisional obviousness-type double patenting rejection because the conflicting.claims have not in fact been patented.

5. Claims 7, 17 and 23 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting over claims 1, 10, 19, 20 and 22 of copending Application No. 09/759695.

Although the conflicting claims are not identical, they are not patentably distinct from each other because they are directed to substantially the same invention and recites only obvious differences which would have been obvious to one of ordinary skill in the art of program development at the time of invention such as simply (i) omitting/adding steps or elements along with their functions, and/or (ii) implementing the method steps with means for performing the steps, and/or (iii) computer program implementation of the method, and/or (iv) implementing a system and product having computer program for performing the method steps, as explained below.

The corresponding claims are as follows:

Instant claim:	copending claims:
7	1, 10, 19, 20 and 22
17	1, 10, 19, 20 and 22
23	1, 10, 19, 20 and 22

Per claim 7:

Copending claim 7 recites a step of **defining an object with defined with an option data structure which supports references to option values without preallocation of memory space for the full option values, as recited in parent claim 1, which corresponds to the step of copending claim 1, and notifying objects of a change in an option value through a change handler identified by an option binding, the option binding being located by first searching a mapping data structure for a previously computed mapping to the option binding and, if no mapping was previously computed, by then computing the mapping to the option binding and storing the mapping in the mapping data structure.**

The copending claim 7 does not explicitly recite the additional limitations, as further recited in instant claim 1:

i) **defining an object with defined fields to support values in preallocated memory space** as recited in instant claim 1, but Official Notice is taken that this step was well known in the art of object oriented program development at the time of invention was made: A class definition typically includes one or more fields that declare and store data of various types. The compiler assigns a memory location (preallocate) to each field in the program based on its type accordingly and then the attribute value of the field is stored in the memory space preallocated to that field -- each instance of a class (object) has its own copy of the fields defined in the class. Therefore, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the step recited in the co-pending claim by adding the step of defining object with defined fields for the purpose of defining and storing data in the memory space preallocated by the compiler. Alternatively, the omission of defined fields from the instant method would be obvious because option values for the purpose of expediting the method.

Ii) **accessing a field value and accessing an option value in the object using expressions of the same syntactic form**, as further recited in instant claim 1. The co-pending claim 7 does not explicitly recite this step. However, both field values and option values are property values of an instance object or a class. Official Notice is taken that this definition was well known in the art of object oriented program development at the time of invention was made: in object oriented programming, properties are usually made as private so that only member methods have access to the properties for purpose of encapsulation. Accessors (getters/setters) are typical of a well-designed object used for accessing a property of a class/object. Regardless of the different memory allocation methods of option values and field values, the use of the same syntax would be obvious because only member methods can access both option and field values. For example, the member method call syntax is objectName.methodName() in Java or C++. The dot operator is used between class/object name and method name: objectName.getValue(); objectName.setValue(field, option); objectName.fieldValue; objectName.optionValue. It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the step recited in the co-pending claim 7 by adding the step of **accessing a field value and accessing an option value in the object using expressions of the same syntactic form,** as recited in instant claim 1 because both option and field values are properties of the object.

Per claim 17:

The rejection of claims 7 is respectively incorporated, and further, the instant claim recites a system corresponding to copending claim 10 and 19, respectively, modified in the manner set forth above in connection with claim 7 respectively. It would have been obvious for one of ordinary skill in the art of program development to implement the copending method modified in the manner set forth above with a system including means for performing the steps of the copending method.

Per claim 23:

The rejection of claims 7 and 17 is respectively incorporated, and further, the instant claim recites a computer product corresponding to copending claim 20, respectively, modified in the manner set forth above in connection with claims 7 and 17 respectively. It would have been obvious for one of ordinary skill in the art of program development to implement the copending method modified in the manner set forth above with a computer product including means for performing the steps of the copending method. The instant claim recites instructions to notify objects of a change in an option value which corresponds to the step of copending claims 20, except that the instant claim does not explicitly recite how to notify objects. It would have been obvious for one of ordinary skill in the art of program development at the time the instant invention was made to modify the copending method by omitting the method of notifying objects in copending claim 20 for the purpose of expediting the method.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

Oath/Declaration

5. The new Oath request has been withdrawn, as applicant's statement is persuasive.

Information Disclosure Statement

6. The applicant requests a copy of the examiner's acknowledgment corresponding to the IDS filed 10/19/2001. However, the examiner is not aware of submission of IDS on 10/19/2001. For consideration, it is respectfully requested to submit the IDS. The information disclosure statements filed 4/13/2005, 12/6/2004, and 2/2/2004 have been considered.

Specification

7. The objection to the specification has been withdrawn due to the amendment to the Specification.

Drawings

8. The drawing filed 12/6/2004 has been acknowledged.

Claim Rejections - 35 USC § 101

9. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claim 25 is rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

Claim 25 is non-statutory because it is directed to a "software system" without recitation of a computer or a computer-readable medium embodying the computer readable instructions. The claim merely recites a "software system" that is disembodied arrangement so as to be called a "computer program" or compilation of facts, information, or data *per se*, without creating any functional interrelationship, either as part of the stored data or as part of the computing processes performed by the computer ("acts") or computer readable medium so as to enable the computer to perform the claimed computer readable instructions as recited. Causing an action or an intended action is different from actually performing an action. "Computer readable instructions" do not necessarily mean that reading instructions by the computer is actually performed. Thus the claim represents non-functional descriptive material that is not capable of producing a useful result, and hence represents only abstract ideas. Therefore, the claim is non-statutory.

Claim Rejections - 35 USC § 102

10. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

11. Claims 1-3, 7, 11-13, 17, 21-23, and 25 are rejected under 35 U.S.C. 102(b) as being anticipated by Berry et al. (U.S. Patent No. 5,732,271, hereinafter "Berry").

As per claim 1, Berry discloses a method and system which provides a prototypical object (object /data structure) that can be copied to create a derived object (See abstract) for processing data comprising: defining an object with defined fields to support values in preallocated memory space (Column 2, Lines 37-41, "The foregoing objects are achieved by a method and system which provides a prototypical object which can be copied to create a derived object. A derived object can contain attribute values or it can hold references to prototypical objects;" Column 3, Lines 51-54, "For example, button object 220 can contain attributes such as background color, button action and size. Rectangle object 214, on the other hand, can contain attributes such as fill color, contents, border type and size.") and with an option data structure which supports references to option values without preallocation of memory space for the full option values (see Column 2, Lines 37-41, "The foregoing objects are achieved by a method and system which provides a prototypical object which can be copied to create a derived object. A derived object can contain attribute values or it can hold references to prototypical objects;" Column 3, Lines 51-54, "For example, button object 220 can contain attributes such as background color, button action and size. Rectangle object 214, on the other hand, can contain attributes such as fill color, contents, border type and size;" The Examiner interprets only attribute values contained in the derived object need memory space preallocated; for those reference it holds do not require memory preallocation (already allocated by the prototypical objects); and accessing a field value stored in one of the defined fields and accessing an option value not stored in the defined fields in the object using expressions of the same syntactic form (Column 2, Lines 37-41, "The foregoing objects are achieved by a method and system which provides a prototypical object which can be copied to create a derived object. A derived object can contain attribute values or it can hold references to prototypical objects;" Column 3, Lines 51-54, "For example, button object 220 can contain attributes such as background color, button action and size. Rectangle object 214, on the other hand, can contain attributes such as fill color, contents, border type and size," see also col 3, lines 39-64; see also the double patenting rejection of claim 1).

As per claim 2, Berry discloses a method as claimed in claim 1 wherein the option data structure identifies change handler code that is executed when an option value changes (Column 2, Lines 41-50, "If a required value is not held by a prototypical object, the present invention discloses a scheme by which the object searches up an object hierarchy to find the required attribute. In addition, each object can register interests in prototypical objects. If an attribute of a prototypical object changes, the prototypical object informs all registered objects of the change. At runtime, the prototypical object becomes a master object whose attribute values can be changed by the user. Changes in master object attributes are propagated to all registered derived objects.").

As per claim 3, the rejection of claim 2 is incorporated and Berry further discloses a method as claimed in claim 2 wherein change handler code for one option is defined in different classes within a class inheritance hierarchy and the change handler code from each class is executed when the option value changes (see Column 2, Lines 41-50, "If a required value is not held by a prototypical object, the present invention discloses a scheme by which the object searches up an object hierarchy to find the required attribute. In addition, each object can register interests in prototypical objects. If an attribute of a prototypical object changes, the prototypical object informs all registered objects of the change. At runtime, the prototypical object becomes a master object whose attribute values can be changed by the user. Changes in master object attributes are propagated to all registered derived objects;" Column 4, Lines 22-31, "FIG. 3 depicts top card 310 of the attribute sheet for button object 220. Button 220 has one attribute: background color. This attribute is indicated by tab 312 associated with the top card 310. To change the background color for button object 220 the developer selects tab 312 on attribute sheet 310, then the developers selects a color from the displayed palette of rectangle objects 316. After a color is selected, the developer can edit the color by pressing button 318, select the color by closing the attribute sheet using window button 324 or cancel the selection by pressing button 320.").

As per claim 7, Berry discloses a method as claimed in claim 1 further comprising: notifying objects of a change in an option value through a change handler identified by an option binding, the option binding being located by first searching a mapping data structure for a previously computed mapping to the option binding and, if no mapping was previously computed, by then computing the mapping to the option binding and storing the mapping in the mapping data structure (see Column 2, Lines 41-50, "If a required value is not held by a prototypical object, the present invention discloses a scheme by which the object searches up an object hierarchy to find the required attribute. In addition, each object can register interests in prototypical objects. If an attribute of a prototypical object changes, the prototypical object informs all registered objects of the change. At runtime, the prototypical object becomes a master object whose attribute values can be changed by the user. Changes in master object attributes are propagated to all registered derived objects.").

As Per Claim 17, the rejection of claim 11 is incorporated respectively and further Berry discloses change handlers which notify objects of a change in an option value and a mapping data structure which maps an option name and class to an option binding which identifies a change handler (Column 2, Lines 41-50, "If a required value is not held by a prototypical object, the present invention discloses a scheme by which the object searches up an object hierarchy to find the required attribute. In addition, each object can register interests in prototypical objects. If an attribute of a prototypical object changes, the prototypical object informs all registered objects of the change. At runtime, the prototypical object becomes a master object whose attribute values can be changed by the user. Changes in master object attributes are propagated to all registered derived objects.").

As per claims 11-13 and 21, they are the system versions of claims 1-3, respectively, and are rejected for the same reasons set forth in connection with the rejection of claims 1-3 above.

As per claims 22-23, they are the product versions of claims 1 and 7, respectively, and are rejected for the same reasons set forth in connection with the rejection of claims 1 and 7 above.

As per claim 25, it is the data signal version of claims 1, 11, 21, and 22, respectively, and is rejected for the same reasons set forth in connection with the rejection of claims 1, 11, 21, and 22 above.

Claim Rejections - 35 USC § 103

12. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 4-6, 8-10, 14-16, 18-20, and 24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Berry et al. (US Patent No. 5,732,271, art of record, hereinafter "Berry"), in view of Hostetter et al., "Curl: A Gentle Slope Language for the Web," World Wide Web Journal, spring, 1997, art of record, hereinafter "Hostetter").

As per claim 4, Berry discloses a method of processing data comprising defining an object with defined fields and with an option data structure accessing a field value and an option value using the same syntactic form. Berry, however, does not explicitly disclose his teaching for using the type description in the option data structure to process an operation on the option value during compilation.

Hostetter discloses the method further comprising: during compilation, using the type description in the option data structure to process an operation on the option value (Page 1, Lines 22-29," **Programming complex operations.** Other components of an interactive document may require more sophisticated mechanisms than are provided by the interface toolkit. These components can also be developed using Curl since, at its heart, Curl is really an object-oriented programming language. **Curl expressions, class definitions and procedure definitions embedded in the web document are securely compiled to native code by the built-in on-the-fly compiler and then executed without the need for any sort of interpreter**. Curl provides many of the features of a modern object-oriented programming language: multiple inheritance, extensible syntax, a strong type system that includes a dynamic "any" type, safe execution through encapsulation of user code and extensive checking performed both at compile and run time. Almost all of the Curl system and compiler are written in Curl;" page 7, lines 11-12, **"Lexically-scoped environment.** Curl provides a structured name space whose bindings include variables, constants, **types, and compilation** hooks for arbitrary syntactic forms."). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hostetter into the method of Berry. The modification would have been obvious because one of ordinary skill in the art would have been motivated to use Curl modern object-oriented programming feature (object structure) to compile to the native code and "execute without the need for any sort of interpreter."

As per claim 5, Berry does not explicitly disclose getting the option value. However, Hostetter teaches an option data structure includes a default value, the method further comprising, in a get operation to an instance of the class, if an option value which applies to the instance has been set, getting the set option value and, if a value which applies has not been set, getting the default value for the class (see Section3, Page 4, Lines 1-2, "The screen shot above reflects the fact the user has selected something **besides the default** color (red) and quantity (0)."). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hostetter into the method of Berry, to get the option value. The modification would have been obvious because one of ordinary skill in the art would have been motivated to incorporate a dynamic object with a simple mechanism for propagating changes in its value to other dynamic objects that depend on first object's value and to customize the object using the option value.

As per claim 6, Berry does not explicitly disclose encoding an option operation. However, Hostetter teaches defining a first class with a first option data structure of a first form which supports, in instances of the class, references to option values without preallocation of memory space for the full option values (see Page 4, Figure 2, item hbox); defining a second class with a second option data structure of a second form which supports, in instances of the second class, references to option values without preallocation of memory space for the full option values, the second form being different from the first form (see Page 4, Figure 2, item vbox) and during compilation, encoding an option operation as a method call to an object of the first class and to an object of the second class without regard to the form of the option data structure supported by the class (see Page 4, Figure 2, item hbox and item vbox; Page 3, Line 20-24, "Since the values for color and quantity are Dynamic objects, the last line of the display changes automatically as the user manipulates the color and quantity controls. A Dynamic object incorporates a simple mechanism for propagating changes in its value to other dynamic objects that depend on first object's value. More sophisticated propagation rules could be supplied by the user by creating a new class of objects derived from Dynamic objects that have a different "propagate" method;" Page 9, Lines 20-22, "Hboxes and vboxes. These are one-dimensional formatters that create simple horizontal or vertical arrangements of their children, lining up their baselines or margins. As in TeX, the relative allocation of white space is controlled by the elasticity of any glue objects that have been added as children"). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hostetter into the method of Berry. The modification would have been obvious because one of ordinary skill in the art would have been motivated to display changes automatically as the user manipulates the color and quantity controls.

As per claim 8, Berry does not explicitly disclose an option list. However, Hostetter teaches that the option data structure comprises a linked list of option items having option values (see Section 4.3, Page 10, Lines 21-22, "Much of the flexibility of boxes comes from the use of properties to control the rendering of primitive objects. A property is a (name, value) binding and each Graphic object has an associated list of properties."). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hostetter into the method of Berry. The modification would have been obvious because one of ordinary skill in the art would have been motivated to implement properties in a dynamically bound environment using a deep binding mechanism and to facilitate the selection of options using option list.

As per claim 9, Berry does not explicitly disclose a nonlocal option value applies to other objects in a nonlocal option hierarchy. However, Hostetter teaches a nonlocal option value applies to other objects in a nonlocal option hierarchy (see Section3, Page 4, Lines 1-2, "The screen shot above reflects the fact the user has selected something besides the default **color** (red) and quantity (0)."). Color is a nonlocal option because all text in a given document is usually the same color.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hostetter into the method of Berry, to comprise a nonlocal option value that applies to other objects in a nonlocal option hierarchy. The modification would have been obvious because one of ordinary skill in the art would have been motivated to implement properties in a dynamically bound environment using a deep binding mechanism.

As per claim 10, Berry does not explicitly disclose that the nonlocal option hierarchy is a graphical hierarchy. However, Hostetter teaches that the nonlocal option hierarchy is a graphical hierarchy *(see* Section3, Page 4, Lines 12, "The screen shot above reflects the fact the user has selected something besides the default color (red) and quantity (0);" Section 4.3, Page 9, Lines 34-35, "text. Properties control the color, size and font family as well as indicating whether the text should be bold or italic.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hostetter into the method of Berry. The modification would have been obvious because one of ordinary skill in the art would have been motivated to represent a graphic image as a hierarchical tree of Graphic objects (Leaves of the tree are primitive Graphic objects which know how to draw themselves, usually after looking up the values of various properties).

<u>As per claim 16</u>, Berry does not explicitly disclose plural classes having data structures of different forms. However, Hostetter teaches plural classes having data structures of different forms, and a compiler which encodes an option operation as a method call to an instance object of one of the classes without regard to the form of the option data structure supported by the class. (See Page 4, Figure 2, item hbox and item vbox; Page 4, Figure 2, item hbox and item vbox; Page 3, Line 20-24, "Since the values for color and quantity are Dynamic objects, the last line of the display changes automatically as the user manipulates the color and quantity controls. A Dynamic object incorporates a simple mechanism for propagating changes in its value to other dynamic objects that depend on first object's value. More sophisticated propagation rules could be supplied by the user by creating a new class of objects derived from Dynamic objects that have a different "propagate" method;" page 9, Lines 20-22, "Hboxes and vboxes. These are one dimensional formatters that create simple horizontal or vertical arrangements of their children, lining up their baselines or margins. As in TeX, the relative allocation of white space is controlled by the elasticity of any glue objects that have been added as children."). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hostetter into the method of Berry, to comprise plural classes having data structures of different forms. The modification would have been obvious because one of ordinary skill in the art would have been motivated to display changes automatically as the user manipulates the color and quantity controls.

As per claims 14, 15, and 18-20, they are the system versions of claims 4, 5, and 8-10, respectively, and are rejected for the same reasons set forth in connection with the rejection of claims 4, 5, and 8-10 above.

As per claim 24, it is the product version of claims 8 and 18, respectively, and is rejected for the same reasons set forth in connection with the rejection of claims 8 and 18 above.

Response to Arguments

13. Applicant's arguments filed 6/24/2004 have been fully considered but they are not persuasive.

Per claims 1, 11, 21, 22, and 25:

The Applicant states that:

Berry does not discuss the claimed technique for defining an object with an option data structure that supports references to option values without preallocation of memory for full option values. It appears that Berry is consistent with traditional object-oriented techniques, as Berry does not suggest modifying them in any way (remark 13).

In response, the claims **do not recite** an option data structure that supports references to option values without preallocation of memory for full option values by "modifying them in any way" and "any technique for avoiding the memory preallocation." Simply stating that the invention does not preallocate memory does not make the claims patentably distinct over the prior art. It appears that the applicant argues that avoiding preallocation is not known in "traditional object-oriented techniques." As the claims do not recite what the option data structure is, the examiner considers it as any data structure that supports dynamic memory allocation such as pointers (an address, a link, or a reference), linked lists, trees etc. Dynamically created memory does not have a name as variables do so we need a reference for it. Once it is initialized, it points to a specific memory location at execution time. Berry states that a "derived object can contain attribute values or it can hold references to prototypical objects (col. 2 lines 37-50)" and "object will search its hierarchy to get the value for that attribute (col. 4 lines 46-60)" while each "prototypical object...maintains a list of all derived objects which have registered an interest in it (col. 4 lines 39-45)." Therefore, the option data structure considered as a dynamic data structure is well known in the object oriented programming environment as taught by Berry. If applicant means anything more, this must be brought out in the claims to further clarify the invention.

In response to the applicant's argument that Berry does not discuss a technique that uses the same syntactic form for accessing values stored in a field or option data structure, the examiner considers the option data structure as a dynamic data structure since the claims do not define what the option data structure is. Accordingly, accessing member values (any variables) of an object is performed using the same existing syntactic form such as object.get_set format applied in Berry. If applicant means anything more, this must be brought out in the claims to further clarify the invention.

* Regarding the rejections based on APA have been withdrawn in view of applicant's amendment to the background section of the specification. Therefore, Applicant's argument regarding these rejections is moot.

** The applicants wish to place the double patenting rejection in abeyance until the claims are otherwise allowable. Therefore, the rejection is maintained above.

Conclusion

14. THIS ACTION IS MADE FINAL. Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

15. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Insun Kang whose telephone number is 571-272-3724. The examiner can normally be reached on M-F 7:30-4 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on 571-272-3719. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

I. Kang
Examiner
6/9/2005

[illegible]

[illegible]

KAKALI CHAKI
ORY PATENT
OLOGY CENT